Topics in Learning Theory

Lecture 8: Kernel Methods (II) and Rule Learning

Topics

- Generalization bound for kernel methods
- Four different kernel representations
- Another nonlinear learning method: decision tree learning

Representations for RKHS Regularizations

- RKHS representation: $\mathcal{H} = \{f(x) : \|f\|_{\mathcal{H}}^2 \leq a^2\}$
- Kernel representation: $f(x) = \sum_{i=1}^{n} \alpha_i k(X_i, x)$, with $\|f\|_{\mathcal{H}}^2 = \alpha^T K_m \alpha \leq a^2$.
- Feature space representation: $f(x) = w^T \psi(x)$, with $\|f\|_{\mathcal{H}}^2 = \|w\|_2^2$

Rademacher Complexity for Kernel Learning

- Rademacher complexity in feature representation:

$$R(\mathcal{H}|S_n) \leq \frac{a}{n}\sqrt{\sum_{i=1}^{n} \|\psi(X_i)\|_{\mathcal{H}}^2}$$

- Equivalent kernel Rademacher complexity:

$$\sum_{i=1}^{n} \|\psi(X_i)\|_{\mathcal{H}}^2 = tr(K_n),$$

thus

$$R(\mathcal{H}|S_n) \leq \frac{a}{n}\sqrt{tr(K_n)}$$

- Data-dependent Rademacher bound for kernel learning: if $\phi \in [0,1]$ with Lipschitz constant $1/\gamma$, then with probability $1-\eta$

$$E_{X,Y}\phi(\hat{f}(X),Y) \leq \frac{1}{n}\sum_{i=1}^{n}\phi(\hat{f}(X_i),Y_i) + \frac{2a}{\gamma n}\sqrt{tr(K_n)} + 3\sqrt{\ln(2/\eta)/(2n)}.$$

L_∞-covering for Kernel Learning

- L_∞-covering in feature representation:

$$\ln N_\infty(\mathcal{H}, \epsilon, n) \leq 36 \frac{a^2 b^2}{\epsilon^2} \ln[2\lceil 4ab/\epsilon + 2 \rceil n + 1],$$

  where $b = \sup_x \|\psi(x)\|_{\mathcal{H}}$.

- Equivalent kernel Rademacher complexity:

$$b = \sup_x \sqrt{k(x,x)}.$$

  note that $tr(K_n) \leq bn$.

Four representations of kernel learning: Least squares regression example in transductive learning setting

- Labeled training data $(x_1, y_1), \ldots, (x_n, y_n)$.
- Unlabeled test data $x_{n+1}, \ldots, x_m$.
- Features $\psi(x_i) \in R^p$
- Kernel $k(x_i, x_j) = \psi(x_i)^T \psi(x_j)$
- Kernel gram matrix $K_{m \times m} = [\psi(x_i)^T \psi(x_j)]_{i,j=1}^m$
- Want to find: $\hat{f} \in R^m$: prediction values on $x_1, \ldots, x_m$.

Primal feature-space formulation (ridge regression)

$$\hat{f}_i = \hat{w}^T \psi(x_i), \quad \hat{w} = \arg\min_w \left[\frac{1}{n} \sum_{i=1}^{n} (w^T \psi(X_i) - Y_i)^2 + \lambda w^T w \right].$$

Solution: $\hat{w} = (\sum_{i=1}^{n} \psi(X_i)\psi(X_i)^T + \lambda n I_{p\times p})^{-1} \sum_i \psi(X_i) Y_i$,

$$\hat{f} = [\psi(X_1), \ldots, \psi(X_m)]^T \left(\sum_{i=1}^{n} \psi(X_i)\psi(X_i)^T + \lambda n I_{p\times p} \right)^{-1} \sum_{i=1}^{n} \psi(X_i) Y_i$$

Primal kernel formulation

$$\hat{f}_i = \sum_{j=1}^{n} k(x_i, x_j)\hat{\alpha}_j, \quad \hat{\alpha} = \arg\min_{\alpha \in R^n} \left[\frac{1}{n}(K_{n\times n}\alpha - Y)^2 + \lambda \alpha^T K_{n\times n}\alpha \right].$$

Solution: $\hat{\alpha} = (K_{n\times n} + \lambda n I_{n\times n})^{-1} Y.$

$$\hat{f} = K_{m\times n} \left(K_{n\times n} + \lambda n I_{n\times n}\right)^{-1} Y$$

Dual kernel formulation

$$\hat{f}_i = \sum_{j=1}^{n} k(x_i, x_j)\hat{\alpha}_j, \quad \hat{\alpha} = \arg\max_{\alpha \in R^n} \left[-\lambda \alpha^T K_n \alpha + 2\lambda \alpha^T Y - \lambda^2 n \alpha^T \alpha \right].$$

Solution: $\hat{\alpha} = (K_{n\times n} + \lambda n I_{n\times n})^{-1} Y$.

$$\hat{f} = K_{m\times n}(K_{n\times n} + \lambda I_{n\times n})^{-1} Y$$

Primal-value $\geq$ Dual-value, and equal at $\hat{\alpha}$: difference $((\lambda + K_n)\alpha - Y)^2$

Primal RKHS (Gaussian processes) formulation

$$\hat{f} = \arg\min_{f \in R^m} \left[\frac{1}{n} \sum_{i=1}^{n} (f_i - Y_i)^2 + \lambda f^T K_m^{-1} f \right].$$

Solution:

$$\hat{f} = \left(\begin{bmatrix} I_{n \times n} & 0 \\ 0 & 0 \end{bmatrix} + \lambda n K_{m \times m}^{-1} \right)^{-1} \begin{bmatrix} Y \\ 0 \end{bmatrix}$$

- Given kernel gram matrix K_m, the RKHS norm of $f \in R^m$ is $f^T K_m^{-1} f$.

Graph Learning

Define regularization condition:

$$f^T K_m^{-1} f = \sum_{(i,j)\in E} (f_i - f_j)^2.$$

- E: edges of a graph on the node
  - usually nearest neigbhor graph: connect nodes (i, j) that are close.
- This regularization condition called (unregularized) graph Laplacian
  - defines a regularization condition (kernel) using both labeled and unlabeled data.
  - encode intuition that if i and j are close, then $f_i \approx f_j$.
  - more general regularization: $\sum_{i,j} w(i, j)(f_i - f_j)^2$.

Summary of Kernel

- Fancy L_2 regularization: learning in infinty-dimensional Hilbert space
  - How to solve the computational problem? through kernel representation.
  - nonlinear in the original input space x.
  - linear in the high-dimensional feature space.
- Can we solve infinity dimensional L_1 regularization problem?
  - through weak-learning + greedy algorithm.
  - nonlinearity introduced in weak-learning.

Non-linear Prediction Rules

- Linear (parameter estimation) model: $\boldsymbol{f(x) = \sum_j w_j \psi_j(x)}$.
  - w_j: unknown parameters to be learned.
  - $\psi_j(x)$: nonlinear basis functions of x.
- Computationally simple.
- How to construct nonlinear basis functions $\psi_j(x)$:
  - hand-crafted functions.
  - kernels of the form $k(\xi_j, x)$: kernel methods.
  - prediction rules
  - ...

Rule Based Classification

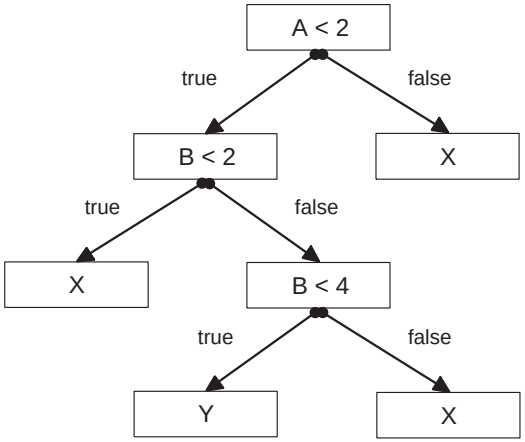
shr $\rightarrow$ earn
div $\rightarrow$ earn
dividend $\rightarrow$ earn
payout $\rightarrow$ earn
qtr $\rightarrow$ earn
earnings & sees $\rightarrow$ earn
quarter & cts $\rightarrow$ earn
split $\rightarrow$ earn
profit $\rightarrow$ earn
OTHERWISE $\rightarrow$ $\sim$earn

Figure 1: Example Decision Rules for Reuters “earn” category

- Easy to understand and possibly modify by a human
- Can incorporate nontext features more easily than other methods

Rule Learning through Decision Trees

Figure 2: Example Decision Tree



- Equivalent rules (read along tree path):

$A < 2$ & $B < 2 \rightarrow$ category-X $A < 2$ & $B \geq 2$ & $B < 4 \rightarrow$ category-Y ...

- Additive model:
  - Each leaf-node is a model
  - tree is an addition of leaf-node models

Decision Trees

- Partition the data into segments along paths to leaf-nodes
- Follow branch at each node through test:
  - is an attribute value $<$ a threshold?
- Constant prediction at each node:

$$\text{probability score} = \frac{\text{number of in-class documents reaching the node}}{\text{number of documents reaching the node}}$$

- Decision tree learning: two-stage process
  - Tree growing: recursively search (attribute,threshold) pair to reduce error
  - Tree pruning: remove deep tree nodes to avoid overfitting

Tree Growing

- Given smooth (convex) loss function $L(f, y)$ (such as $(f-y)^2$) and n training data (X_i, Y_i) $(i = 1, \ldots, n)$

- Recursively do the following:
  - at each leaf-node, let S be the training data reaching it
  - the optimal loss at the node is: $\min_f \sum_{i \in S} L(f, Y_i)$.
  - for each partition (attribute,threshold) pair (j, θ),
    - partition S into $S_1(j, \theta)$ and $S_2(j, \theta)$ using the test
    - the optimal loss with this partition $\min_{f_1, f_2}[\sum_{i \in S_1} L(f_1, Y_i) + \sum_{i \in S_2} L(f_2, Y_i)]$
  - for each leaf node: grow the tree by using (j, θ) that reduces the loss most

- Stopping criteria:

  - Depth-first: A certain depth is reached.
  - Best-first: each time split the node with the best loss reduction, until a fixed number of nodes is reached.

- Numerical versus categorical attributes:
  - numerical: ordered
  - categorical: unordered — each split can partition into arbitrary subsets

- Missing data:
  - put as an extra value
  - use zero value
  - imputation assuming missing at random.

Example loss criteria

- Least squares (regression tree):

$$\hat{f}_1 = \sum_{i \in S_1} Y_i / |S_1|, \quad \hat{f}_2 = \sum_{i \in S_2} Y_i / |S_2|.$$

$$\Delta L = \sum_{i \in S_1} (Y_i - \hat{f}_1)^2 + \sum_{i \in S_2} (Y_i - \hat{f}_2)^2.$$

- Least squares (classification tree) with $Y_i = 0, 1$:
  - Min loss (Gini-index):

$$Q(S) = \min_f \sum_{i \in S} w_i (f - Y_i)^2 = W(S) p(S)(1 - p(S)).$$

- $W(S) = \sum_i w_i$.
- let $p(S) = (\sum_{i \in S} w_i Y_i)/W(S) = P(Y = 1|S)$.

- Log-loss (classification tree) with $Y_i = 0, 1$:

$$Q(S) = \min_f \sum_{i \in S} w_i(-Y_i \ln \hat{f}_j - (1 - Y_i) \ln \hat{f}_j)$$

We have $\hat{f}_j = p(S_j)$ and

$$Q(S) = -w(S)[p(S) \ln p(S) + (1 - p(S)) \ln(1 - p(S))].$$

- General for classification: purity measure $Q(p)$ ($p \in [0, 1]$), and split according to

$$W(S)Q(p) - \sum_{j=1}^{2} W(S_j)Q(p(S_j)).$$

$Q(p)$ is a symmetric function of $p - 0.5$ and strictly concave — $p \approx 0$ or 1 are pure.

- How about 0-1 loss?
  - $Q(p) = 0.5 - |p - 0.5|$.
  - not good for greedy search.
  - $p(S_1 + S_2) = 0.7 \rightarrow p(S_1) = 0.5, p(S_2) = 0.9$ does not indicate progress.

- Example: $0.7 \rightarrow [0.5, 0.9]$ and $0.7 \rightarrow [0.6, 0.8]$.

Tree Pruning

- Fully grown tree tends to overfit the data
  - data are partitioned into very small segments
  - insufficient data at each leaf node to reliably estimate probability
- Pruning: removing deep tree nodes so that leaf nodes in the resulting tree contain sufficient data for reliable probability estimate
  - many different methods
- Prune to a fixed tree size:
  - given a loss function $L'(f, y)$
  - loss may differ from training loss: e.g. non-smooth classification loss

– recursively removing leaf-nodes with least reduction of loss L' until number of leaf-nodes reaches a fixed size

Complexity of Decision Tree

- Let T be the depth of the tree, then under appropriate assumptions

$$R(\mathcal{H}|n) \propto T/\sqrt{n}.$$

Remarks on Decision Tree

- Advantages:
  - interpretable
  - handle non-homogeneous features easily
  - finds non-linear interactions
- Disadvantage:
  - usually not the most accurate classifier by itself